Studying the effect of server side constraints on the makespan of the no-wait flow shop problem with sequence dependent setup times

Hamed Samarghandi
Department of Finance and Management Science, Edwards School of Business,
University of Saskatchewan, Saskatoon, Saskatchewan, Canada, S7N 5A7
samarghandi@edwards.usask.ca

Abstract

This paper deals with the problem of scheduling the no-wait flow shop system with sequence dependent setup times and server side constraints. No-wait constraints state that there should be no waiting time between consecutive operations of jobs. In addition, sequence dependent setup times are considered for each operation. This means that the setup time of an operation on its respective machine is dependent on the previous operation on the same machine. Moreover, the problem consists of server side constraints, i.e., not all machines have a dedicated server to prepare them for an operation. In other words, several machines share a common server. The considered performance measure is makespan. This problem is proved to be strongly NP-Hard. To deal with the problem two genetic algorithms (GA) are developed. In order to evaluate the performance of the developed frameworks, a large number of benchmark problems are selected and solved with different server limitation scenarios. Computational results confirm that both of the proposed algorithms are efficient and competitive. The developed algorithms are able to improve many of the best-known solutions of the test problems from the literature. Moreover, the effect of the server side constraints on the makespan of the test problems is explained using the computational results.

Keywords: Flow Shop Scheduling; No-wait; Sequence Dependent Setup; Makespan; Server Side Constraints; Genetic Algorithm

1. Introduction

The no-wait flow shop problem is a special case of the classical flow shop problem, in which there should be no waiting time between successive operations of jobs. In other words, once processing is started, no interruption is permitted between operations of the same jobs. This paper also considers a

sequence dependent setup time for each operation. Therefore, setup time of a machine for a specific operation depends on the previous operation that is processed on that machine. Allahverdi *et al.* (1999) and Aldowaisan (2001) describe the importance of considering setup times in no-wait scheduling problems. In this paper, server side constraints are also considered, meaning that a number of machines are assigned a single server that is responsible for performing the setup operations on all of these machines. As a result, setup times on the machines with a common server should not overlap. Companies always look for ways to reduce waste and improve efficiency; therefore, reducing the number of servers can be used in different situations. Whether the server is a robot or a human, reducing the number of servers without sacrificing efficiency is desirable.

The considered performance measure is makespan. Following the three-field notation of the scheduling problems, the mentioned problem can be designated as $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$. King and Spachis (1980) proved that the no-wait flow shop problem with makespan performance measure ($F \mid no-wait \mid C_{\max}$) can be transformed to the Asymmetric Travelling Salesperson Problem (ATSP). Röck (1984) proved that ($F \mid no-wait \mid C_{\max}$) is NP-Hard. Aldowaisan (2001) transformed the no-wait flow shop problem with separable setup times and the makespan criterion ($F \mid no-wait, setup \mid C_{\max}$) to ATSP. Since $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ is a generalization of $F \mid no-wait \mid C_{\max}$ and $F \mid no-wait, setup \mid C_{\max}$, it can be inferred that $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ is also strongly NP-Hard.

Sequence dependent setup times occur in many practical instances. Examples of such circumstances include (Samarghandi and ElMekkawy 2014a):

- Adjusting jigs and fixtures for processing different products.
- Retooling multi-tool machines.
- Cleaning machines to make them ready for the next operation. Cleaning is an indispensable part of the manufacturing processes in industries such as textile, plastic, chemical, semi-conductor, pharmaceutical, and food industries.

Industrial applications mentioned in the literature for $F,S_Q \mid no-wait,S_{sd} \mid C_{\max}$ include chemical industries (Rajendran 1994), food industries (Hall and Sriskandarajah 1996), steel production (Wismer 1972), pharmaceutical industries (Raaymakers and Hoogeveen 2000), and production of concrete products (Grabowski and Pempera 2000). Hall and Sriskandarajah (1996) provide a comprehensive review of the applications of the problem.

In this paper, a Genetic Algorithm (GA) as well as a GA with diversified local search procedure are developed to deal with $F,S_Q \mid no-wait,S_{sd} \mid C_{\max}$. Moreover, an algorithm is developed to create a feasible timetable from a given sequence. The timetabling algorithm is further coupled with the developed genetic algorithms to explore the feasible region of the problem.

Although $F,S_Q \mid no-wait,S_{sd} \mid C_{\max}$ has numerous practical applications, it has received no attention in the literature. In this paper, different server limitation scenarios are considered, and computational results are compared with the results of the 2-Opt algorithm. Moreover, computational results are compared with the most competitive methods for $F \mid no-wait,S_{sd} \mid C_{\max}$ from the literature. In fact, $F \mid no-wait,S_{sd} \mid C_{\max}$ can be considered as a special case of $F,S_Q \mid no-wait,S_{sd} \mid C_{\max}$.

The contribution of this paper is three-fold. First, $F,S_Q \mid no-wait,S_{sd} \mid C_{\max}$ is studied in this paper for the first time; a mathematical model is developed for this problem and a number of small-instance test problems are solved to optimality. Second, the effect of adding server side constraints with different scenarios on the makespan of $F \mid no-wait,S_{sd} \mid C_{\max}$ is studied by applying the developed GAs to a large number of test problems. Finally, although the algorithm is developed to deal with sequence dependent setup times and server constraints, it outperforms competitive methods specifically designed for $F \mid no-wait,S_{sd} \mid C_{\max}$. Computational results show that the proposed GA methods are able to find good-quality solutions for the test problems in a reasonable time. It is hoped that the presented results will be used as a benchmark by other researchers interested in solving similar scheduling problems in the future.

The rest of the paper is outlined as follows. Section 2 performs a literature review. Section 3 is devoted to problem description. Section 4 explains the proposed GA methods. Section 5 summarizes the computational results. Section 6 discusses the concluding remarks and future research directions.

2. Literature Review

The first attempts to deal with the no-wait flow shop problem should be credited to Reddi and Ramamoorthy (1972), Wismer (1972), Grabowski and Syslo (1973), Bonney and Gundry (1976), King and Spachis (1980), Gangadharan and Rajendran (1993), Rajendran (1994), Glass et al. (1999), Sidney et al. (2000), and Sviridenko (2003). The two-machine no-wait flow shop problem with setup and removal times was reduced to the famous Travelling Salesperson Problem (TSP) by Gupta et al. (1997). Cheng et al. (1999) studied the problem of $F2,S1|setup|c_{\max}$ and proposed some heuristics for the problem. Bianco et al. (1999) proposed two heuristics for the no-wait flow shop problem with release dates and sequence dependent setup times, and makespan criterion.

Aldowaisan and Allahverdi (1998) considered $F2|no-wait,setup|\sum C_i$ and developed a heuristic algorithm for the problem. Aldowaisan (2001) performed a research on the same problem and developed a new heuristic algorithm. In addition, Aldowaisan and Allahverdi (2004) proposed six heuristics for $F|no-wait|c_{\max}$ and considered the separable setup time in the problem of $F|no-wait|\sum C_i$.

Sidney et al. (2000) considered the two-machine no-wait flow shop problem with anticipatory setup times and makespan and proposed a heuristic for this problem. Guirchoun et al. (2005) studied a two-stage hybrid flow shop with no-wait constraint between the two stages and proposed a heuristic to deal with the problem. Grabowski and Pempera (2005) proposed 6 meta-heuristics for $F|no-wait|C_{\max}$. Liu et al. (2007) proposed a particle swarm optimization with several local search approaches. Su and Lee (2008) considered the problem of two-machine no-wait flow shop with separable setup times and single server and developed a heuristic and a branch-and-bound algorithm to

solve the problem. Laha and Chakraborty (2009) proposed a constructive algorithm for $F|no-wait|C_{\max}$.

The literature on $F|no-wait,setup|C_{\max}$ and $F|no-wait,S_{sd}|C_{\max}$ is rather limited. Problems of $F2,S1|no-wait,setup|C_{\max}$ and $J2,S1|no-wait,setup|C_{\max}$ have been studied by Samarghandi and ElMekkawy (2011) and Samarghandi and ElMekkawy (2013a) respectively. $F,S_Q|no-wait,S_{sd}|C_{\max}$ should be considered as a generalization of $F2,S1|no-wait,setup|C_{\max}$. Moreover, Samarghandi and ElMekkawy (2014a) studied the problem of $F|no-wait,S_{sd}|C_{\max}$. Computational results of Samarghandi and ElMekkawy (2014a) will be used to perform several comparisons with the results of the developed algorithms in this paper. Table 1 summarizes the available literature on the subject of this paper.

Table 1- Literature review

Research	Problem Considered	Proposed Method
Gupta *et al.* (1997)	Two-machine flow shop problem with setup and removal times	Reduction to TSP
Cheng *et al.* (1999)	$F2,S1 \mid setup \mid C_{\max}$	Heuristic
Bianco *et al.* (1999)	$F \mid no-wait, release \mid C_{\max}$	Heuristic
Aldowaisan and Allahverdi (1998)	$F2 \mid no-wait, setup \mid \sum C_i$	Heuristic
Sidney *et al.* (2000)	Two-machine no-wait flow shop problem with anticipatory setup times and makespan	Heuristic
Aldowaisan (2001)	$F2 \mid no-wait, setup \mid \sum C_i$	Heuristic
Aldowaisan and Allahverdi (2004)	$F \mid no-wait \mid C_{\max}$	Heuristic
Guirchoun *et al.* (2005)	Two-stage hybrid flow shop with no-wait constraint between the two stages	Heuristic
Grabowski and Pempera (2005)	$F \mid no-wait \mid C_{\max}$	Various meta-heuristic
Liu *et al.* (2007)	$F \mid no-wait \mid C_{\max}$	PSO
Su and Lee (2008)	$F2 \mid no-wait, setup \mid \sum C_i$	Branch and bound
Qian *et al.* (2009)	$F \mid no-wait \mid C_{\max}$	Hybrid differential evolution
Pan *et al.* (2008a)	$F \mid no-wait \mid C_{\max}$	Hybrid PSO
Pan *et al.* (2008b)	$F \mid no-wait \mid C_{\max}$	Greedy algorithms
Laha and Chakraborty (2009)	$F \mid no-wait \mid C_{\max}$	Constructive heuristic
Araujoa and Naganoa (2011)	$F \mid no-wait, S_{sd} \mid C_{\max}$	Heuristic
Samarghandi and ElMekkawy (2011)	$F2,S1 \mid no-wait, setup \mid C_{\max}$	Hybrid variable neighbourhood search
Samarghandi and ElMekkawy (2012a)	$F \mid no-wait \mid C_{\max}$	Hybrid tabu search
Samarghandi and ElMekkawy (2012b)	$F \mid no-wait, setup \mid C_{\max}$	PSO and genetic algorithm
Nagano *et al.* (Article in Press)	$F \mid no-wait, setup \mid \sum C_i$	Evolutionary clustering search
Gao *et al.* (2012)	$F \mid no-wait, setup \mid \sum C_i$	Hybrid harmony search
Jolai *et al.* (Article in Press)	No-wait flexible flow shop scheduling problem with sequence dependent setup times	Several metaheuristics
Rabiee *et al.* (Article in Press)	No-wait two-machine flow shop problem with sequence dependent setup times and probable rework	Several metaheuristics
Ying *et al.* (2012)	No-wait flow shop manufacturing cell scheduling problem (FMCSP) with sequence dependent family setup times	Several metaheuristics

3. Problem Description

3.1. Notations and Mathematical Model

The following notation is used throughout this paper:

M	Set of machines
$m = \|M\|$	Number of machines
N	Set of jobs
$n = \|N\|$	Number of jobs
J_i	Job i
o_{ij}	j th operation of J_i
p_{ij}	Processing time of the j th operation of J_i on its respective machine
S_i	Starting time of J_i
$S_{o_{ij}}$	Starting time of o_{ij}
ST_{ijk}	Setup time of o_{ij} if scheduled after o_{kj}
ST_{ij0}	Setup time of o_{ij} if J_i is the first scheduled job
$S_{ST_{ij}}$	Starting time of the setup time of o_{ij}
$SR_w \subseteq M$ $w = 1,2,...,Q$	A subset of M which includes the machines with one assigned server
$\|SR_w\|$	Number of members of SR_w
π_l	Sequence l
$C_{\max}$	Makespan of π_l

Brackets are used to indicate consecutive jobs, i.e., $S_{[i]}$ refers to the starting time of the job planned to be operated after i th job in a given sequence. Moreover, suppose that $SR_w; w = 1,2,...,Q$ is a subset of M and presents the set of machines for which one server is assigned to perform the setups. If Q servers exist in a particular instance of the problem ($w = 1,2,...,Q$), then $SR_w \cap SR_e = \phi; 1 \le w < e \le Q$. In other words, it is assumed that each machine is assigned to only one server. Based on the above notations, a mathematical model for $F, S_Q \,|\, no-wait, S_{sd} \,|\, C_{\max}$ is as follows:

$$Min \ C_{\max} \tag{1}$$

$$C_{\max} \ge S_{o_{im}} + p_{im}; \qquad i = 1,2,...,n \tag{2}$$

$$S_{o_{[i]j}} \geq S_{o_{ij}} + p_{ij} + ST_{[i]ji}; \quad i = 1,2,...,n-1 \quad j = 1,2,...,m \tag{3}$$

$$S_{o_{i[j]}} = S_{o_{ij}} + p_{ij}; \quad i = 1,2,...,n \quad j = 1,2,...,m-1 \tag{4}$$

$$S_{ST_{[i]j}} - S_{ST_{[i]k}} \geq ST_{[i]ki}; \quad i = 1,2,...,n-1 \quad \forall j,k \in SR_w; j > k \quad w = 1,2,...,Q \tag{5}$$

$$S_{ST_{1j}} - S_{ST_{1k}} \geq ST_{1k0}; \quad \forall j,k \in SR_w; j > k \quad w = 1,2,...,Q \tag{6}$$

$$S_{o_{ij}} \geq 0; \quad i = 1,2,...,n \quad j = 1,2,...,m \tag{7}$$

$$S_{ST_{ij}} \geq 0; \quad i = 1,2,...,n \quad j = 1,2,...,m \tag{8}$$

In this model, the objective function is to minimize the makespan. (2) guarantees that the makespan is equal to the completion time of the last scheduled operation. (3) indicates that the starting time of the j th operation of $[i]$ (or the job scheduled after i) should not be before the starting time of the same operation of i plus its processing time plus the setup time of $o_{[i]j}$ when its previous operation (i in this case) is taken into consideration. (4) imposes the no-wait constraints. (5) represents the server side constraints for all the jobs except the first job scheduled in the sequence. Server side constraints for the first job in the sequence is represented by (6). Finally, (7) and (8) set the non-negativity constraints. Figure 1 illustrates the implication of (5). In this figure, it is assumed that one server is assigned to machines j and k.

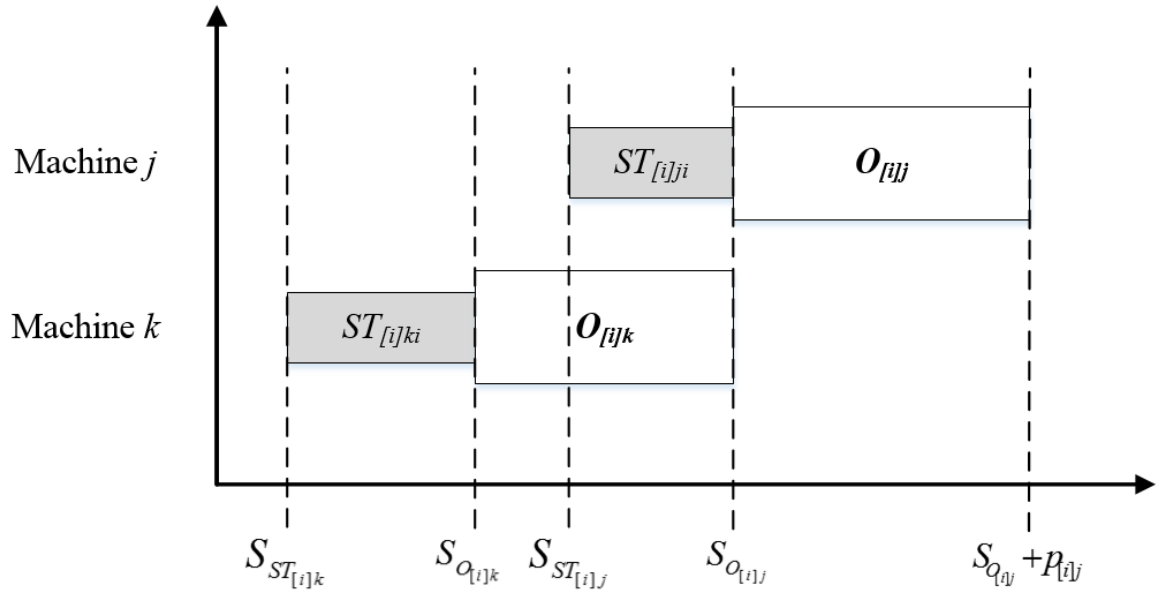

Figure 1 – Illustration of the Mathematical Model

Based on figure 1, one can verify that:

$$S_{ST_{[i]j}} - S_{ST_{[i]k}} \geq ST_{[i]ki} \tag{9}$$

If (9) is violated, then setup times of $o_{[i]k}$ and $o_{[i]j}$ overlap, which is a violation of server constraints. According to (4), once the starting time of o_{i1} is obtained by the model, it is possible to calculate the starting time of $o_{ij}; j = 2,3,...,m$. In other words, it is possible to reduce the problem to finding the best time to start $o_{i1}; i = 1,2,...,n$ without violating server side constraints. Consequently, $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ can be reduced to the Asymmetric Travelling Salesperson Problem (ATSP).

3.2. Calculating the Makespan

An algorithm is developed here in order to calculate the objective function of $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ by generating a feasible timetable from a given sequence of jobs. This algorithm is called Makespan Calculation Algorithm with Server constraints or MCAS.

MCAS utilizes a pointer (e) and a dereference operator, denoted as $h(e)$. A pointer refers to the place of an element in a set. For instance, if $SR_w = \{3,6,9,10\}$, then $e = 2$ points to the element that is located in the second place in SR_w. The dereference operator shows the element that the pointer has referred to. Therefore, if $e = 2$, then $h(e) = 6$. MCAS calculates the makespan of a given permutation π_l from $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$.

To schedule the first job of π_l:

1. Set $w \leftarrow 1$.
2. Sort the indices of SR_w in the ascending order; suppose that $|SR_w| = b$. Define e as the pointer of SR_w. Set $e \leftarrow 1$. Set $S_{ST_{1,h(e)}} = 0$.
3. Set $e \leftarrow e + 1$.

4. Set $S_{ST_{1,h(e)}} = S_{ST_{1,(h(e-1))}} + ST_{1,h(e),0}$.

5. If $e < b$, go back to step 3. If $e = b$ and $w < Q$, set $w \leftarrow w+1$ and go to step 2. If $e = b$ and $w = Q$, proceed to step 6.

6. Set $S_{o_{11}} = S_{ST_{1,1}} + ST_{1,1,0}$.

7. For $k = 1$ to $m-1$, $S_{O_{1[k]}} = \max\{S_{ST_{1,[k]}} + ST_{1[k]0}, S_{O_{1k}} + p_{1k}\}$. If $S_{ST_{1,[k]}} + ST_{1[k]0} > S_{O_{1k}} + p_{1k}$, set $d = S_{ST_{1,[k]}} + ST_{1[k]0} - (S_{O_{1k}} + p_{1k})$, and for $z = 1,2,...,k-1$, set $S_{O_{1z}} \leftarrow S_{O_{1z}} + d$.

   To schedule the remaining jobs of π_l:

8. Set $i \leftarrow 1; j \leftarrow 1$.

9. Set $w \leftarrow 1$.

10. Sort the indices of SR_w in the ascending order; suppose that $|SR_w| = b$. Define e as the pointer of SR_w. Set $e \leftarrow 1$. Set $S_{ST_{[i],h(e)}} = S_{o_{i,h(e)}} + p_{i,h(e)}$.

11. Set $e \leftarrow e+1$.

12. Set $S_{ST_{[i],h(e)}} = \max\{S_{ST_{[i],(h(e-1))}} + ST_{[i],h(e-1),i}, S_{O_{i,h(e)}} + p_{i,h(e)}\}$.

13. If $e < b$, go back to step 11. If $e = b$ and $w < Q$, set $w \leftarrow w+1$ and go to step 10. If $e = b$ and $w = Q$, proceed to step 14.

14. Set $S_{o_{[i]j}} = S_{ST_{[i]ji}} + ST_{[i]ji}$.

15. $j \leftarrow j+1$.

16. $S_{O_{[i]j}} = \max\{S_{ST_{[i]j}} + ST_{[i]ji}, S_{O_{[i],(j-1)}} + p_{[i],(j-1)}\}$. If $S_{ST_{[i]j}} + ST_{[i]ji} > S_{O_{[i],(j-1)}} + p_{[i],(j-1)}$, set $d = S_{ST_{[i]j}} + ST_{[i]ji} - (S_{O_{[i],(j-1)}} + p_{[i],(j-1)})$, and for $z = 1,2,..., j-1$, set $S_{O_{[i]z}} \leftarrow S_{O_{[i]z}} + d$.

17. If $j < m$, go back to step 15. Otherwise, proceed to step 18.

18. If $i = n$, stop. $C_{\max} = S_{o_{nm}} + p_{nm}$. Otherwise, set $i \leftarrow i+1$ and $j = 1$. Go back to step 9.

MCAS starts with a sequence of jobs (π_l) or equivalently, a permutation. MCAS first schedules the sequence dependent setup times of the first job in π_l based on the defined server side constraints (steps 1 to 5). Then the no-wait constraints are imposed, while modifying the starting time of the rest of the operations of that job if necessary (steps 6 and 7). When scheduling the first job of π_l is completed, using the same method, MCAS first schedules the sequence dependent setup times of the next job and imposes the server constraints (steps 8 to 13). Steps 14 to 16 schedule the operations and impose the no-wait constraints. Finally, step 17 calculates the makespan and then the algorithm is completed. Computational complexity of this algorithm is $O(mn)$.

3.3. Illustrative Example

Table 2 presents the data for a typical instance of $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$. For this example, all setup times are assumed to be equal to 1, except $ST_{2,1,1} = 3$ and $ST_{2,2,1} = 2$. Suppose that $SR_1 = \{1,2\}$ and $SR_2 = \{3\}$. $\pi = (1,2,3)$ is considered as the desired sequence and MCAS will be used to develop a timetable.

Table 2 – A Typical $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ Instance

J_i	p_{ij}		
1	1	2	1
2	1	1	3

The first 5 steps of MCAS schedule the setup times of the first job in π according to the server constraints. According to step 2, MCAS sets $S_{ST_{1,1}} = 0$. Since $SR_1 = \{1,2\}$, in step 4 MCAS sets $S_{ST_{1,2}} = S_{ST_{1,1}} + ST_{1,1,0} = 0+1 = 1$. Moreover, since $SR_2 = \{3\}$, which means machine 3 has its dedicated server, MCAS sets $S_{ST_{1,3}} = 0$. Figure 2 depicts the partial timetable developed so far.

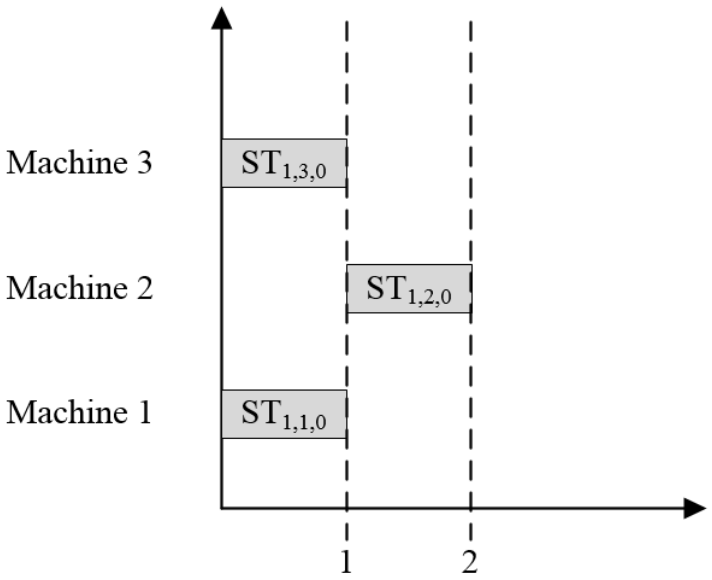

Figure 2 – Setup Times of J_1

At this point MCAS proceeds to steps 6 and 7. In step 6, MCAS sets $S_{o_{11}} = S_{ST_{1,1}} + ST_{1,1,0} = 0+1=1.$ Step 7 develops the following set for $k=1$: $S_{O_{1[k]}} = \max\{S_{ST_{1,[k]}} + ST_{1[k]0}, S_{O_{1k}} + p_{1k}\} \rightarrow S_{O_{1,[1]}} = S_{O_{1,2}} = \max\{1+1,1+1\} = 2$. The same calculation results in $S_{O_{1,[2]}} = S_{O_{1,3}} = \max\{S_{ST_{1,[2]}} + ST_{1,[2],0}, S_{O_{1,2}} + p_{1,2}\} = \max\{0+1,2+2\} = 4$ for $k=2$. Thus, so far the Gantt chart of figure 3 is developed.

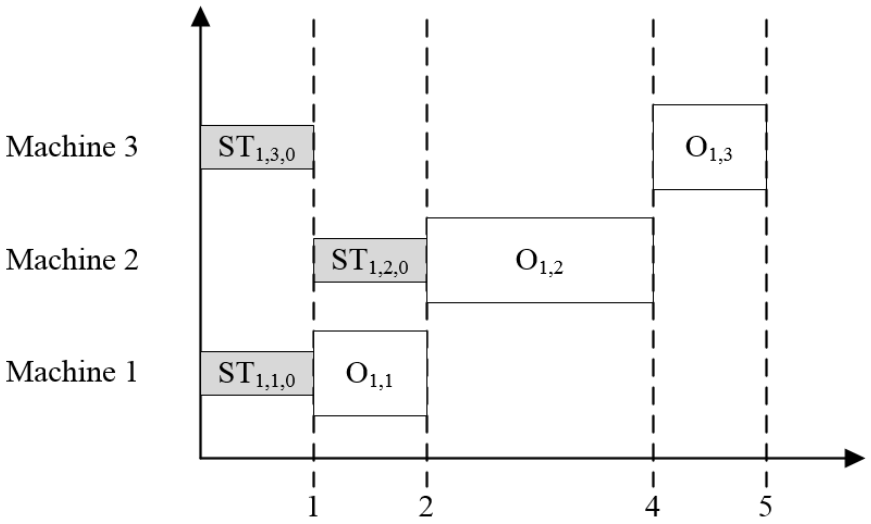

Figure 3 – Job 1 is Scheduled

Steps 8 to 13 schedule the setup times of the second job in sequence π the same way that setup times of job 1 are scheduled. A partial Gantt chart after scheduling the setup times of job 2 is presented

in figure 4. Since there is one server assigned to machines 1 and 2, their setup times should not overlap. However, machine 3 has its own dedicated server and therefore its setup times can overlap with setup times of machines 1 and 2.

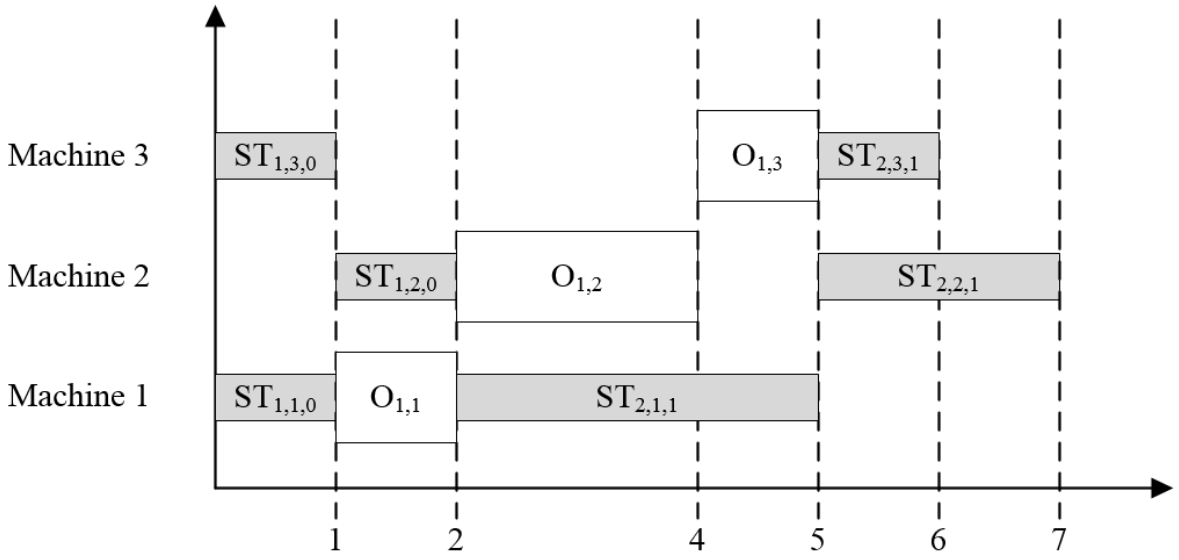

Figure 4 – Partial Gantt Chart After Scheduling Setup Times of Job 2

Then, MCAS proceeds to step 14 and since $i=1; j=1,$ sets $S_{O_{[i]j}} = S_{ST_{[i]ji}} + ST_{[i]ji} \rightarrow S_{O_{[1],1}} = S_{ST_{[1],1,1}} + ST_{[1],1,1} = 2+3=5$. Step 15 sets $j=2$ and MCAS proceeds to step 16. At this step:

$$S_{O_{[i]j}} = \max\{S_{ST_{[i]j}} + ST_{[i]ji}, S_{O_{[i],(j-1)}} + p_{[i],(j-1)}\} \rightarrow$$
$$S_{O_{[1],2}} = S_{O_{2,2}} = \max\{S_{ST_{2,2}} + ST_{2,2,1}, S_{O_{2,1}} + p_{2,1}\} = \quad (10)$$
$$\max\{5+2, 5+1\} = 7$$

The same calculations for $j=3$ results in:

$$S_{O_{[i]j}} = \max\{S_{ST_{[i]j}} + ST_{[i]ji}, S_{O_{[i],(j-1)}} + p_{[i],(j-1)}\} \rightarrow$$
$$S_{O_{[1],3}} = S_{O_{2,3}} = \max\{S_{ST_{2,3}} + ST_{2,3,1}, S_{O_{2,2}} + p_{2,2}\} = \quad (11)$$
$$\max\{5+1, 7+1\} = 8$$

Therefore, the Gantt chart of Figure 5 is developed.

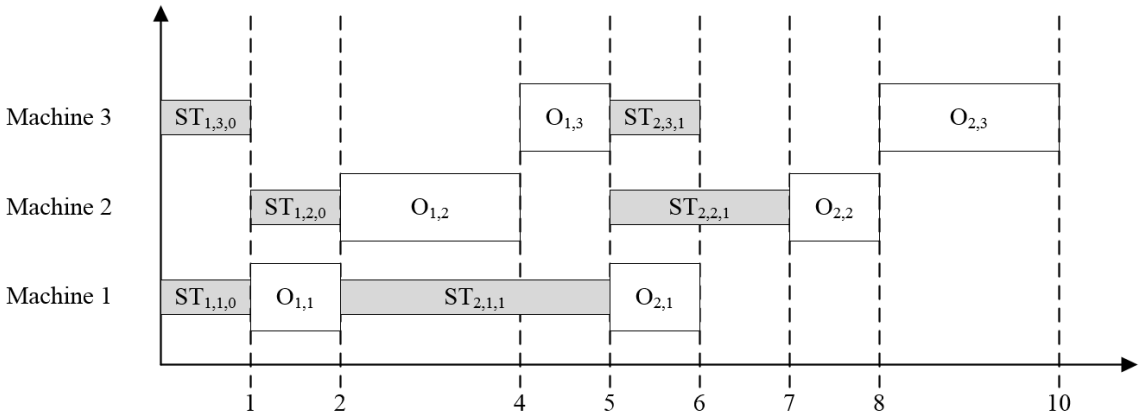

Figure 5 – Gantt Chart Before Imposing No-Wait Constraints

Based on step 16, for $j = 2$ one can verify that:

$$\begin{aligned} &S_{ST_{[i]j}} + ST_{[i]ji} > S_{O_{[i],(j-1)}} + p_{[i],(j-1)} \rightarrow \\ &S_{ST_{[1],2}} + ST_{[1],2,1} > S_{O_{[1],1}} + P_{[1],1} \rightarrow \\ &5 + 2 > 5 + 1 \end{aligned} \tag{12}$$

In other words, MCAS verifies that no-wait constraints are violated. Therefore, step 16 performs extra steps to impose this constraint:

$$\begin{aligned} d &= S_{ST_{[i]j}} + ST_{[i]ji} - (S_{O_{[i],(j-1)}} + p_{[i],(j-1)}) \\ &= S_{ST_{[1],2}} + ST_{[1],2,1} - (S_{O_{[1],1}} + p_{[1],1}) \\ &= 5 + 2 - (5 + 1) \\ &= 1 \end{aligned} \tag{13}$$

And $d = 1$ will be added to $S_{O_{[1],1}}$:

$$S_{O_{[1],1}} \leftarrow S_{O_{[1],1}} + d = 5 + 1 = 6 \tag{14}$$

This results in the Gantt chart of figure 6. At this point, the algorithm is finished and a complete timetable is created; step 18 of MCAS returns $C_{\max} = S_{O_{2,3}} + p_{2,3} = 8 + 2 = 10$ as the makespan of π. The proposed solution methodology is explained in the next section.

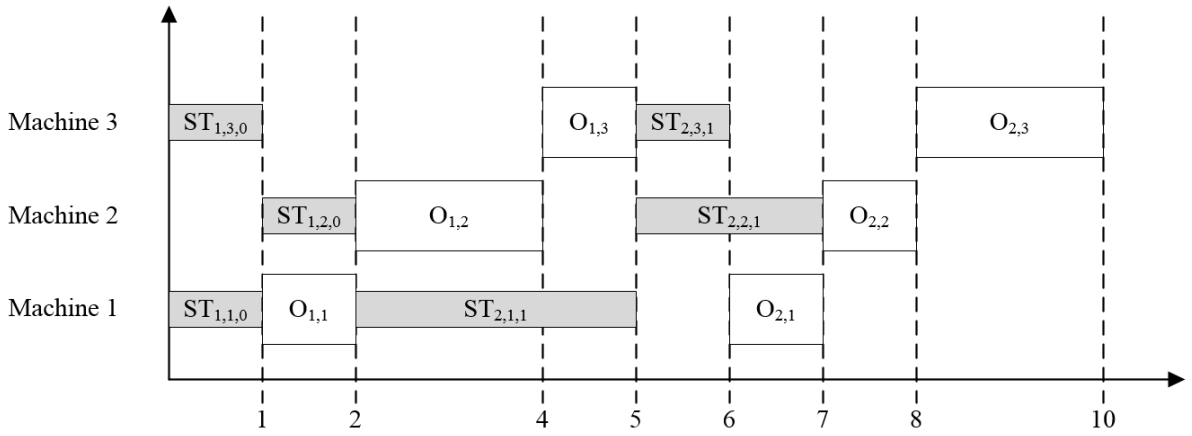

Figure 6 – Final Gantt Chart After Imposing No-Wait Constraints

4. The Proposed Genetic Algorithm

Genetic Algorithm (GA) is the main search technique in this paper. GAs are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. GA uses chromosomes to code the feasible solutions of the problem. Feasible solutions of $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ are sequences of jobs, denoted by π in section 3. GA is a popular search technique with several successful implementations for the continuous and discrete optimization problems in the literature (Samarghandi and Eshghi 2009, Samarghandi *et al.* 2010, Samarghandi and Jahantigh 2011, Samarghandi and ElMekkawy 2013b, Samarghandi and ElMekkawy 2014b).

4.1. Chromosome Structure and GA Operations

Chromosome structure (genotype) is one of the most important aspects of the genetic algorithm. In the proposed GA, each permutation or sequence of jobs (π) is a chromosome. MCAS generates complete and feasible timetables for $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ once a sequence of jobs is given. This approach defines the extraction of solutions from chromosomes (phenotype). It is worthwhile to mention that the proposed GA uses the operations defined by Shadrokh and Kianfar (2007).

The proposed GA generates Pop random permutations for the first generation. Then, MCAS calculates the makespan of each of these permutations. Calculated makespans will be used as the fitness label of the permutations. Pop is an even number and a parameter of the algorithm, which remains unchanged during all of the iterations of the algorithm. New generations are made from the existing generation, using four operations: crossover, mutation, immigration, and local search.

In the crossover operation, the existing generation is randomly partitioned into $\frac{Pop}{2}$ pairs of parents, and the crossover operation is performed on each pair with probability P_c. If a pair is not

selected for crossover, each individual in this pair is considered for the mutation operation with probability P_m and then for local search with probability P_l.

The crossover operation on a pair of parents, P_1 and P_2, produces two children, C_1 and C_2. Let $f(I)$ be the makespan of schedule I. If $r_i.(f(P_i)+f(C_i)) \le f(P_i)$ then C_i will be selected for local search with probability P_l and then goes to a new generation and P_i dies out ($i=1,2$). r_i is a random number generated from the interval $[0,1]$ for each i. Otherwise C_i dies out and P_i is considered for mutation with probability P_m and then for local search with probability P_l. It should be noted that $r_i.(f(P_i)+f(C_i)) \le f(P_i)$ determines how much advancement in the quality of genes should be expected in consecutive generations. However, since r_i is a random number, the amount of gene progress differs in each iteration. Afterwards, the immigration operation is also performed before finalizing the cycle of producing a new generation. Immigration operation feeds the gene pool with randomly generated genes, helps maintain the gene diversity, and helps prevent immature convergence.

For the immigration operation, a chromosome will be randomly generated and is called NEW. An individual I is selected randomly from the current population. Let the probability of leaving I be $P_{Leave}(I, NEW) = \dfrac{f(I)}{f(I)+f(NEW)}$. A random number is generated from the interval $[0,1]$. If this random number is less than $P_{Leave}(I, NEW)$, NEW replaces I; otherwise NEW is discarded. The immigration operation is able to bring new and desirable characteristics to the next gene pools. The chromosome with the best makespan value in the final generation is the result given by the algorithm. Pop, P_s, P_m, and P_l are adjustable parameters of the algorithm. The number of iterations of the proposed GA is another parameter of the algorithm and is denoted as $Iter$.

4.2. Crossover

The proposed GA uses a one-point crossover. Suppose that $P_1 = (J_1^1, J_2^1, ..., J_n^1)$ and $P_2 = (J_1^2, J_2^2, ..., J_n^2)$ are the two individuals that are selected for crossover. The one-point crossover selects an integer number $r \in [1, n]$. Then, the crossover operation is performed and the result is C_1 and C_2 whose chromosomes are defined as $(J_1^{c_1}, ..., J_r^{c_1}, J_{r+1}^{c_1}, ..., J_n^{c_1})$ and $(J_1^{c_2}, ..., J_r^{c_2}, J_{r+1}^{c_2}, ..., J_n^{c_2})$. $(J_1^{c_1}, ..., J_r^{c_1}) = (J_1^1, ..., J_r^1)$ and $J_a^{c_1} = J_b^2; a = r+1, ..., n$ where b is the lowest index such that $J_b^2 \notin \{J_1^{c_1}, ..., J_{a-1}^{c_1}\}$. And $(J_1^{c_2}, ..., J_r^{c_2}) = (J_1^2, ..., J_r^2)$ and $J_a^{c_2} = J_b^1; a = r+1, ..., n$ where b is the lowest index such that $J_b^2 \notin \{J_1^{c_2}, ..., J_{a-1}^{c_2}\}$. The explained one-point crossover operation when $r = 3$ is demonstrated by (15).

$$P_1: 2,3,4 \mid 6,5,1 \rightarrow C_1 : 2,3,4,1,6,5$$
$$P_2: 3,4,1 \mid 6,2,5 \rightarrow C_2 : 3,4,1,2,6,5 \quad (15)$$

4.3. Mutation

Let $P = (J_1, J_2, ..., J_n)$ be the selected chromosome for mutation. Then, the algorithm generates two integer numbers $r_1, r_2 \in [1, n-1]$ and an integer number $a \in [0,1]$. If $a \geq 0.5$, then the new chromosome will be $P_{new} = (J_1, ..., J_{r_1}, J_{r_2}, ..., J_n, J_{r_1+1}, ..., J_{r_2-1})$, while if $a < 0.5$, the new chromosome will be $P_{new} = (J_{r_1+1}, ..., J_{r_2}, J_1, ..., J_{r_1}, J_{r_2+1}, ..., J_n)$. The mutation operation when $n = 9; r_1 = 3; r_2 = 7$ is demonstrated by (16).

$$a \geq 0.5: 1,2,3,4,5,6,7,8,9 \rightarrow 1,2,3,7,8,9,4,5,6$$
$$a < 0.5: 1,2,3,4,5,6,7,8,9 \rightarrow 4,5,6,7,1,2,3,8,9 \quad (16)$$

4.4. Local Search

Once an individual is selected for local search, the algorithm randomly selects two genes from the chromosome and exchanges the places of these genes in the sequence. If the fitness function of the chromosome is improved as a result of this exchange, it will be accepted and the new chromosome will be transferred to the new gene pool. Otherwise, the two genes will be moved back to their original places and the local search procedure will restart. This process can be repeated several times until a solution is ultimately improved. However, in order to maintain the computational efficiency of the proposed GA, the number of iterations of the local search algorithm will be limited to 5. In other words, if the local search algorithm is unable to improve the fitness function of a particular chromosome after 5 attempts, this chromosome will not be transferred to the next gene pool.

4.5. Final Intensification

Once the best makespan and its corresponding sequence of the jobs are selected as the final solution by the GA, a final intensification procedure is performed. This sub-algorithm exchanges the location of the first two adjacent jobs in the sequence and evaluates the makespan of the sequence using MCAS. If the makespan of the new sequence is improved by the exchange, it will be accepted and the exchange sub-procedure will be restarted. If this exchange does not improve the fitness function of the sequence, the exchanged jobs will be moved back to their original locations in the sequence and the next two adjacent jobs in the sequence will be exchanged.

4.6. Genetic Algorithm with Diversified Local Search Procedure

This algorithm follows all of the explained procedures of the developed GA; however, in order to make the GA algorithm more effective, the local search procedure of this algorithm employs different

operations to facilitate a move from a certain solution to an improved solution. The local search algorithm starts with the exchange operation explained in section 4.4. If this operation is not successful after 5 attempts, the algorithm tries the exchange-3 operation. Accordingly, the algorithm randomly selects 3 genes from the chromosome and performs an exchange-3 operation. Suppose that the selected genes are i, j, and k. The exchange-3 operation is described by (17).

$$(1,2,...,i,...,j,...,k,...,n) \xrightarrow{Exchange-3} (1,2,...,k,...,i,...,j,...,n) \quad (17)$$

If the exchange-3 operation is successful, the new chromosome will be transferred to the new gene pool; otherwise, the 3 genes will be moved back to their original locations. The number of exchange-3 attempts before the local search moves to the next operation is 5. The last operation that the local search algorithm will apply to a chromosome is called a sectional swap, which will also be applied to a chromosome for a maximum of 5 times until either an improved chromosome is found or the unimproved chromosome is discarded. For the sectional swap operation, a gene in the chromosome is randomly selected. Suppose that the selected gene is i. The sectional swap procedure is defined by (18).

$$(1,2,...,i,i+1,...,n) \xrightarrow{Sectional\ Swap} (i+1,i+2,...,n,1,2,...,i) \quad (18)$$

In order to distinguish between the GA algorithm with diversified local search procedure and the GA algorithm with simple local search procedure, the former will be called GA+DLS, while the latter will simply be called GA throughout the rest of this paper. The pseudo code of the GA+DLS method is as follows:

1. Generate Pop random permutations to initiate the first gene pool. Calculate the fitness of each individual chromosome with the MCAS algorithm.
2. Partition the chromosomes to $\frac{Pop}{2}$ pairs. Apply the cross over operation to each pair with probability P_c.
3. If a pair is not selected for cross over, apply the mutation operation to each individual in this pair with probability P_m.

4. Candidate the remaining chromosomes for the local search procedure with probability P_l.

   4.1. Start with exchange procedure and if unsuccessful, repeat this approach for 5 times. If the exchange sub-algorithm results in an improved solution, proceed to step 5; otherwise, go to step 4.2.

   4.2. Apply the exchange-3 algorithm to the permutation and repeat for 5 times if unsuccessful. If the exchange-3 method results in an improved solution, proceed to step 5; otherwise, go to step 4.3.

   4.3. Apply the sectional swap approach to the chromosome and repeat for 5 times if unsuccessful. Proceed to step 5.

5. Calculate the fitness of all of the newly generated solutions with MCAS algorithm and create the next gene pool.

6. Repeat steps 2 to 5 for $Iter$ iterations.

7. Perform the final intensification procedure to the best solution found and return the resulting chromosome as the final solution of the algorithm.

The next section presents the computational results.

5. Computational Results

5.1. Tuning Parameters

As seen in section 4, the developed GA has 5 parameters that must be tuned before the search can be started. Sensitivity analysis has been performed to determine the effect of the different values of these parameters on the performance of the algorithm. Accordingly, 3 different problems from the literature were chosen: rec01+SD ($m=5, n=20$), rec25+SD ($m=30, n=15$), and rec35+SD ($m=50, n=10$); each problem was considered with two different server constraints as described by (19) and (20).

$$\begin{cases} |SR_i| = 2; \forall i \in \left\{1,2,...,\dfrac{n}{2}\right\}; n = 2b; b \in N \\ |SR_i| = 2; \forall i \in \left\{1,2,...,\left\lfloor \dfrac{n}{2} \right\rfloor\right\}; \left|SR_{\left\lfloor \frac{n}{2} \right\rfloor+1}\right| = 1; n = 2b+1; b \in N \end{cases} \quad (19)$$

$$\begin{cases} |SR_i| = 3; \forall i \in \left\{1,2,...,\dfrac{n}{3}\right\}; n = 3b; b \in N \\ |SR_i| = 3; \forall i \in \left\{1,2,...,\left\lfloor \dfrac{n}{3} \right\rfloor\right\}; \left|SR_{\left\lfloor \frac{n}{3} \right\rfloor+1}\right| = 1; n = 3b+1; b \in N \\ |SR_i| = 3; \forall i \in \left\{1,2,...,\left\lfloor \dfrac{n}{3} \right\rfloor\right\}; \left|SR_{\left\lfloor \frac{n}{3} \right\rfloor+1}\right| = 2; n = 3b+2; b \in N \end{cases} \quad (20)$$

It should be noted that, according to (19), the algorithm assigns a dedicated server to the last machine if the number of machines are odd. For instance, if the test problem has 5 machines, then $SR_1 = \{1,2\}; |SR_1| = 2$ and $SR_2 = \{3,4\}; |SR_2| = 2$. However, machine 5 will be assigned one dedicated server; in other words $SR_3 = \{5\}; |SR_3| = 1$. With the same logic, depending on the number of machines, it is possible to have one or two machines instead of three machines assigned to one server, when equation set (20) is in effect. For simplicity, the described conditions of (19) and (20) will be denoted as SR^2 and SR^3 throughout the rest of this paper. The proposed GA algorithm was applied to each problem 3 times with 4 different combinations of the parameter values as follows:

Table 3 – Different Parameter Combinations

Parameter	Combination 1	Combination 2	Combination 3	Combination 4
Pop	$\frac{n}{6}$	$\frac{n}{5}$	$\frac{n}{2}$	n
P_m	0.2	0.3	0.4	0.5
P_c	0.2	0.3	0.4	0.5
P_l	0.05	0.1	0.15	0.2
$Iter$	$10n$	$50n$	$100n$	$200n$

Table 4 presents the resulting makespans for the different combinations of the parameters considered.

Table 4 – Results of the Sensitivity Analyses

Problem	Replication	SR^2				SR^3			
		Parameter Combination				Parameter Combination			
		1	2	3	4	1	2	3	4
rec01+SD	1	2183	2159	2131	2126	2180	2171	2132	2120
	2	2159	2151	2124	2124	2178	2164	2120	2132
	3	2156	2131	2126	2126	2178	2157	2132	2120
rec25+SD	1	4742	4724	4670	4662	4672	4676	4666	4666
	2	4741	4726	4669	4662	4676	4680	4666	4669
	3	4741	4729	4662	4695	4673	4680	4669	4666
rec35+SD	1	6395	6317	6139	6139	6350	6239	6148	6148
	2	6247	6325	6162	6139	6341	6229	6148	6148
	3	6307	6206	6139	6182	6349	6254	6169	6188

Analysis of variance (ANOVA) can be utilized to select the best combination from the considered parameter combinations. Considered factors in the ANOVA include parameter combination as defined by table 3, problem set, server constraints, and the interactions between the mentioned factors. In the mentioned ANOVA, each factor has 3 replications. Table 5 summarizes the results of the ANOVA ($\alpha = 0.05$).

Table 5 - Analysis of Variance for Makespan

Source	Degree of Freedom	Sequential Sums of Squares	Adjusted Sums of Squares	Adjusted Mean Square Value	F-Value	P-Value
Problem	2	203814815	203814815	101907407	193904.16	0
Parameter	3	102481	102481	34160	65	0
SR	1	953	953	953	1.81	0.184
Problem*Parameter	6	51672	51672	8612	16.39	0
Problem*SR	2	4898	4898	2449	4.66	0.014
Parameter*SR	3	2035	2035	678	1.29	0.288
Problem*Parameter*SR	6	7349	7349	1225	2.33	0.047
Error	48	25227	25227	526		
Total	71	204009429				

According to the $p-values$ of table 5, server side constraints are not an important factor in the analysis. Therefore, a re-specification of ANOVA is necessary. Table 6 presents the results of the re-specified model. The importance of the server constraints will be discussed with more details in the following sections.

Table 6 – Results of the Re-Specified ANOVA Model

Source	Degree of Freedom	Sequential Sums of Squares	Adjusted Sums of Squares	Adjusted Mean Square Value	F-Value	P-Value
Problem	2	203814815	203814815	101907407	151116.97	0
Parameter	3	102481	102481	34160	50.66	0
Problem*Parameter	6	51672	51672	8612	12.77	0
Error	60	40462	40462	674		
Total	71	204009429				
	$R^2 = 99.98\%$	$R^2_{adj} = 99.98\%$				

In order to confirm that the analysis of variance presented in table 6 is valid, residuals should follow a normal distribution. Figure 7 illustrates the normal probability plot of the residuals. To conclude that the residuals follow a normal distribution, they should be close to the normal probability line. Figure 7 confirms that the residuals are very close to the normal line.

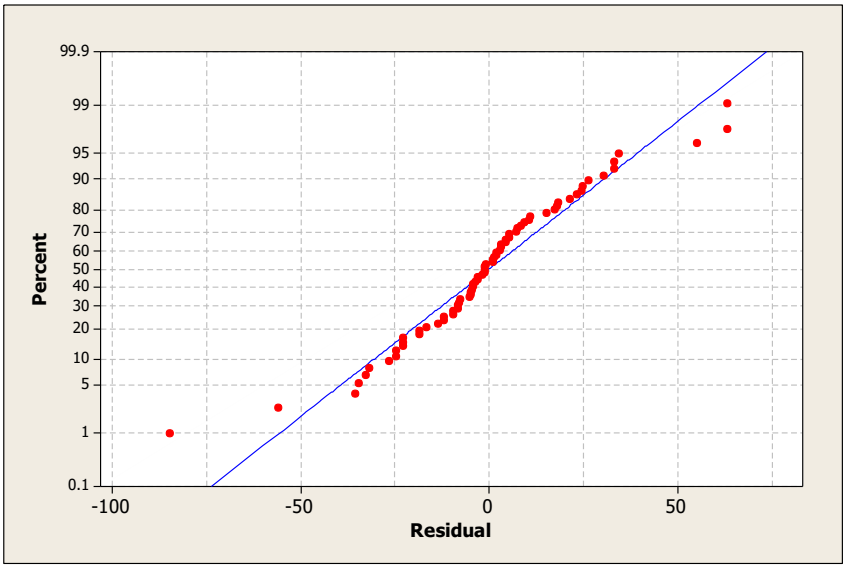

Figure 7 - Normal Probability Plot of the Residuals

As table 6 indicates, the combinations of table 3 have an actual effect on the makespan of the studied test problems. In order to find the best combination among the 4 combinations, the main effects plot proves to be useful. Figure 8 illustrates the main effects plot.

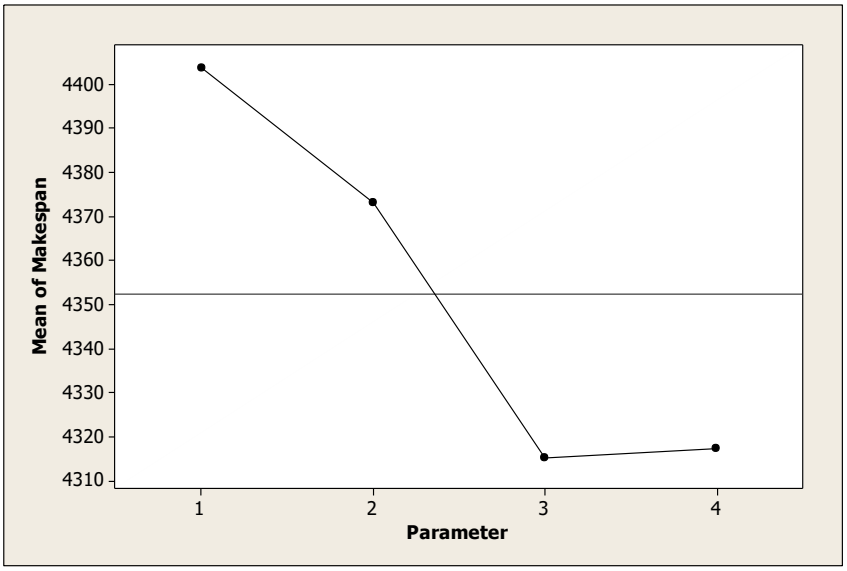

Figure 8 - Main Effects Plot

Figure 8 demonstrates that combinations 3 and 4 of table 3 are more desirable than combinations 1 and 2. Since the difference between combinations 3 and 4 is negligible, and combination 3 requires less computational effort, which leads to less CPU time, this combination is chosen for tuning the parameters of both GA and GA+DLS to perform the computational analysis.

The developed algorithms were coded using Microsoft Visual C++ 2008; all the computational experiments were performed on a PC equipped with a 2.66GHz Intel Pentium IV CPU and 4 GB of RAM.

To test the efficiency of the proposed algorithms, a set of 29 problems were chosen from the literature: car01 through car08 introduced by Carlier (1978) and rec01 through rec41 introduced by Reeves (1995). Reeves (1995) found this specific set of problems difficult to solve. Moreover, optimal solutions for the no-wait version of these problems are unknown. All of these test problems are available at OR-Library (Beasley). Samarghandi and ElMekkawy (2014a) generated sequence dependent setups for the problems of Carlier (1978) and Reeves (1995). These problems were named as car+SD and rec+SD, and solved by a PSO algorithm that was designed for $F \mid no-wait, S_{sd} \mid C_{\max}$. Since $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ is a generalization of $F \mid no-wait, S_{sd} \mid C_{\max}$, car+SD and rec+SD problems along with server constraints of equations (19) and (20) will be used as test problems in this

research. To remain consistent with the literature, each problem is solved 20 times, and the best obtained objective function value as well as the average and worst objective function values are reported. In addition, the average CPU time to obtain the makespans in seconds and standard deviation of the obtained makespans are stated. Section 5.2 reports the computational results of car01 through car08; computational results for rec01 through rec41 appear in section 5.3 and 5.4.

5.2. Computational Results Obtained for car01 through car08

Table 7 presents the computational results of car01+SD through car08+SD. These problems generally have a lower number of jobs compared to the set of rec+SD problems. As a result, it is possible to solve many of them to optimality by means of the mathematical model of section 3. One can verify that the proposed algorithms are in most cases able to produce the optimal solutions. Tables 8 and 9 report more details about the obtained makespans. Table 8 belongs to SR^2 and Table 9 demonstrates the results for SR^3.

5.3. Computational Results of rec01+SD through rec41+SD

Table 10 compares the computational results of the developed algorithms with the makespans generated by the 2-Opt algorithm for problems rec01+SD through rec41+SD. This table considers the case of SR^2. It can be verified that both of the developed algorithms are very efficient, with GA+DLS being slightly better than GA. Small values of the STD column is another indicator of the consistency of the proposed PSO. Table 11 performs the same comparison for the case of SR^3. Section 5.4 compares the results of the developed algorithms for $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ with the results of Samarghandi and ElMekkawy (2014a) for $F \mid no-wait, S_{sd} \mid C_{\max}$.

5.4. Comparison of the Solutions of $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ with $F \mid no-wait, S_{sd} \mid C_{\max}$

Table 12 performs a comparison between the results of Samarghandi and ElMekkawy (2014a) for $F \mid no-wait, S_{sd} \mid C_{\max}$ and the results of the developed frameworks of this paper.

Table 7 – Computational Results for Problems with Optimal Solution

Problem	n,m	No Server Constraint	Optimal Solution for SR^2	Optimal Solution for SR^3	GA - SR^2		GA+DLS - SR^2		GA - SR^3		GA+DLS - SR^3	
		OFV*	OFV	OFV	Best Solution	Gap**	Best Solution	Gap	Best Solution	Gap	Best Solution	Gap
Car01+SD	**11,5**	10,379.00	10,379.00	10,402.00	**10,379.00**	100.000	**10,379.00**	100.000	**10,402.00**	100.000	**10,402.00**	100.000
Car02+SD	**13,4**	N/A	N/A	N/A	11,486.00	N/A	11,486.00	N/A	11,488.00	N/A	11,488.00	N/A
Car03+SD	**12,5**	11,877.00	11,877.00	11,919.00	**11,877.00**	100.000	**11,877.00**	100.000	**11,919.00**	100.000	**11,919.00**	100.000
Car04+SD	**14,4**	N/A	N/A	N/A	12,384.00	N/A	12,384.00	N/A	12,398.00	N/A	12,398.00	N/A
Car05+SD	**10,6**	11,945.00	12,068.00	12,266.00	**12,068.00**	100.000	**12,068.00**	100.000	12,270.00	100.033	**12,266.00**	100.000
Car06+SD	**8,9**	12,015.00	12,131.00	12,131.00	**12,131.00**	100.000	**12,131.00**	100.000	**12,131.00**	100.000	**12,131.00**	100.000
Car07+SD	**7,7**	9,795.00	9,815.00	9,795.00	**9,815.00**	100.000	**9,815.00**	100.000	**9,795.00**	100.000	**9,795.00**	100.000
Car08+SD	**8,8**	11,525.00	11,525.00	11,684.00	**11,525.00**	100.000	**11,525.00**	100.000	**11,684.00**	100.000	**11,684.00**	100.000

* Objective Function Value

** $\frac{OFV_{\text{Algorithm}}}{OFV_{\text{Optimal}}} \times 100$, smaller gaps are more desirable

Table 8 – Detailed Computational Results for the Case of SR^2

Problem	n,m	2-Opt OFV*	GA						GA+DLS					
			Best OFV	Average OFV	Worst OFV	STD**	CPU Time	Gap***	Best OFV	Average OFV	Worst OFV	STD	CPU Time	Gap
car01+SD	**11,5**	14,830.00	**10,379.00**	10,401.10	10,647.00	58.76	8.14	69.99%	**10,379.00**	10,437.33	10,909.00	60.14	10.58	69.99%
car02+SD	**13,4**	15,290.00	**11,486.00**	11,584.65	11,674.00	62.51	8.44	75.12%	**11,486.00**	11,654.55	11,936.00	61.78	10.97	75.12%
car03+SD	**12,5**	15,898.00	**11,877.00**	11,984.55	12,174.00	92.76	8.41	74.71%	**11,877.00**	11,985.88	12,346.00	65.39	10.93	74.71%
car04+SD	**14,4**	16,571.00	**12,384.00**	12,616.05	12,865.00	146.48	8.33	74.73%	**12,384.00**	12,561.98	12,930.00	82.89	10.66	74.73%
car05+SD	**10,6**	15,383.00	**12,068.00**	12,109.55	12,546.00	129.49	8.44	78.45%	**12,068.00**	12,138.15	12,552.00	69.60	10.80	78.45%
car06+SD	**8,9**	15,623.00	**12,131.00**	12,255.60	12,590.00	179.07	9.93	77.65%	**12,131.00**	12,281.18	12,590.00	103.29	12.71	77.65%
car07+SD	**7,7**	12,579.00	**9,815.00**	9,834.80	9,944.00	42.90	7.42	78.03%	**9,815.00**	9,837.20	9,944.00	35.55	9.50	78.03%
car08+SD	**8,8**	14,099.00	**11,525.00**	11,545.70	11,606.00	26.78	8.53	81.74%	**11,525.00**	11,543.23	11,597.00	20.19	10.91	81.74%
Average	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**92.34**	**8.45**	**76.30%**	**N/A**	**N/A**	**N/A**	**62.35**	**10.88**	**76.30%**

* Objective Function Value

**Standard Deviation

*** $\frac{Best\,OFV_{\text{Algorithm}}}{OFV_{\text{2-Opt}}} \times 100$, smaller gaps are more desirable

Table 9 – Detailed Computational Results for the Case of SR^3

Problem	n,m	2-Opt OFV*	GA						GA+DLS					
			Best OFV	Average OFV	Worst OFV	STD**	CPU Time	Gap***	Best OFV	Average OFV	Worst OFV	STD	CPU Time	Gap
car01+SD	**11,5**	13,111.00	**10,402.00**	10,429.60	10,542.00	40.99	8.34	79.34%	**10,402.00**	10,417.58	10,623.00	43.57	10.93	79.34%
car02+SD	**13,4**	15,212.00	**11,488.00**	11,558.45	11,695.00	66.93	8.13	75.52%	**11,488.00**	11,618.53	12,023.00	113.00	10.64	75.52%
car03+SD	**12,5**	17,733.00	**11,919.00**	12,016.90	12,080.00	57.55	8.42	67.21%	**11,919.00**	12,067.90	12,360.00	103.06	11.19	67.21%
car04+SD	**14,4**	15,784.00	**12,398.00**	12,588.75	12,905.00	156.56	8.50	78.55%	**12,398.00**	12,596.18	12,846.00	124.55	11.31	78.55%
car05+SD	**10,6**	15,550.00	12,270.00	12,395.15	12,559.00	112.06	8.54	78.91%	**12,266.00**	12,305.65	12,558.00	65.59	11.02	78.88%
car06+SD	**8,9**	15,814.00	**12,131.00**	12,358.95	12,590.00	233.88	9.69	76.71%	**12,131.00**	12,226.45	12,590.00	162.99	12.49	76.71%
car07+SD	**7,7**	12,846.00	**9,795.00**	9,799.70	9,889.00	21.02	7.49	76.25%	**9,795.00**	9,811.45	9,889.00	36.17	9.59	76.25%
car08+SD	**8,8**	14,607.00	**11,684.00**	11,703.45	11,857.00	39.83	8.66	79.99%	**11,684.00**	11,727.78	12,022.00	86.29	11.09	79.99%
Average	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**91.10**	**8.47**	**76.56%**	**N/A**	**N/A**	**N/A**	**91.90**	**11.03**	**76.56%**

* Objective Function Value

**Standard Deviation

*** $\frac{Best\,OFV_{\text{Algorithm}}}{OFV_{\text{2-Opt}}} \times 100$, smaller gaps are more desirable

Table 10 – Detailed Computational Results for the Case of SR^2

Problem	n,m	2-Opt OFV*	GA						GA+DLS					
			Best OFV	Average OFV	Worst OFV	STD**	CPU Time	Gap***	Best OFV	Average OFV	Worst OFV	STD	CPU Time	Gap
rec01+SD	**20,5**	2,822.00	2,124.00	2,144.15	2,183.00	14.28	11.82	75.27%	**2,118.00**	2,145.58	2,183.00	16.51	15.12	75.05%
rec03+SD	**20,5**	2,824.00	1,911.00	1,929.30	1,973.00	14.53	11.99	67.67%	**1,884.00**	1,930.15	1,987.00	27.14	15.35	66.71%
rec05+SD	**20,5**	2,902.00	**2,007.00**	2,046.80	2,090.00	22.71	11.72	69.16%	2,010.00	2,041.70	2,090.00	18.25	15.00	69.26%
rec07+SD	**20,10**	3,764.00	2,649.00	2,682.85	2,764.00	32.68	19.25	70.38%	**2,637.00**	2,682.33	2,741.00	30.36	24.63	70.06%
rec09+SD	**20,10**	3,655.00	**2,660.00**	2,695.80	2,757.00	21.44	19.22	72.78%	2,675.00	2,695.03	2,741.00	15.49	24.60	73.19%
rec11+SD	**20,10**	3,205.00	2,571.00	2,586.50	2,632.00	16.59	19.41	80.22%	**2,565.00**	2,588.55	2,617.00	13.32	23.48	80.03%
rec13+SD	**20,15**	4,387.00	**3,315.00**	3,352.15	3,437.00	26.68	25.43	75.56%	3,324.00	3,355.25	3,401.00	15.43	30.76	75.77%
rec15+SD	**20,15**	4,330.00	3,253.00	3,277.85	3,322.00	22.75	25.87	75.13%	**3,237.00**	3,272.80	3,328.00	26.12	31.30	74.76%
rec17+SD	**20,15**	4,219.00	3,274.00	3,309.15	3,361.00	28.24	25.10	77.60%	**3,271.00**	3,291.43	3,337.00	17.61	30.36	77.53%
rec19+SD	**30,10**	5,401.00	3,867.00	3,887.25	3,917.00	15.97	32.64	71.60%	**3,851.00**	3,901.63	3,971.00	29.04	39.49	71.30%
rec21+SD	**30,10**	4,980.00	3,743.00	3,776.95	3,824.00	23.54	32.77	75.16%	**3,714.00**	3,748.75	3,832.00	29.48	43.26	74.58%
rec23+SD	**30,10**	5,507.00	3,623.00	3,652.30	3,730.00	32.36	31.37	65.79%	**3,587.00**	3,664.15	3,725.00	26.15	41.40	65.14%
rec25+SD	**30,15**	6,094.00	**4,662.00**	4,708.85	4,768.00	32.03	43.47	76.50%	4,644.00	4,695.03	4,763.00	33.08	57.38	76.21%
rec27+SD	**30,15**	6,348.00	4,562.00	4,611.65	4,673.00	27.17	42.99	71.87%	**4,550.00**	4,589.60	4,620.00	16.33	56.75	71.68%
rec29+SD	**30,15**	6,172.00	4,443.00	4,483.90	4,524.00	24.38	42.47	71.99%	**4,424.00**	4,475.20	4,544.00	32.69	56.05	71.68%
rec31+SD	**50,10**	8,919.00	5,936.00	6,029.60	6,129.00	51.98	100.63	66.55%	**5,911.00**	6,021.28	6,176.00	64.09	127.90	66.27%
rec33+SD	**50,10**	8,917.00	6,159.00	6,211.90	6,352.00	45.00	99.48	69.07%	**6,155.00**	6,225.10	6,294.00	30.13	126.44	69.03%
rec35+SD	**50,10**	9,329.00	**6,139.00**	6,251.05	6,395.00	73.33	100.31	65.81%	6,143.00	6,196.53	6,314.00	39.41	127.49	65.85%
rec37+SD	**75,20**	15,841.00	10,985.00	11,076.60	11,177.00	45.60	604.32	69.35%	**10,957.00**	11,059.95	11,191.00	61.88	742.70	69.17%
rec39+SD	**75,20**	16,783.00	**11,299.00**	11,401.05	11,503.00	68.17	597.69	67.32%	11,303.00	11,450.05	11,701.00	93.80	734.55	67.35%
rec41+SD	**75,20**	16,428.00	11,494.00	11,579.60	11,692.00	65.39	605.11	69.97%	**11,461.00**	11,528.53	11,623.00	42.58	743.67	69.77%
Average	**NA**	**NA**	**NA**	**NA**	**NA**	**33.56**	**119.19**	**71.65%**	**NA**	**NA**	**NA**	**32.33**	**147.99**	**71.45%**

* Objective Function Value

**Standard Deviation

*** $\frac{Best\,OFV_{\text{Algorithm}}}{OFV_{\text{2-Opt}}} \times 100$, smaller gaps are more desirable

Table 11 – Detailed Computational Results for the Case of SR^3

Problem	n,m	2-Opt OFV*	GA						GA+DLS					
			Best OFV	Average OFV	Worst OFV	STD**	CPU Time	Gap***	Best OFV	Average OFV	Worst OFV	STD	CPU Time	Gap
rec01+SD	**20,5**	2,880.00	**2,120.00**	2,144.35	2,180.00	18.75	11.88	73.61%	2,125.00	2,148.60	2,188.00	17.08	15.22	73.78%
rec03+SD	**20,5**	2,826.00	1,890.00	1,916.30	1,950.00	17.02	12.64	66.88%	**1,880.00**	1,922.05	1,965.00	21.59	16.19	66.53%
rec05+SD	**20,5**	2,663.00	2,032.00	2,057.60	2,110.00	23.92	13.03	76.30%	**2,010.00**	2,048.20	2,129.00	24.09	16.69	75.48%
rec07+SD	**20,10**	3,609.00	**2,671.00**	2,711.75	2,752.00	20.04	19.99	74.01%	**2,671.00**	2,698.95	2,772.00	22.48	26.18	74.01%
rec09+SD	**20,10**	3,392.00	**2,644.00**	2,674.35	2,714.00	16.70	19.34	77.95%	2,669.00	2,691.08	2,739.00	17.77	25.34	78.69%
rec11+SD	**20,10**	3,538.00	2,600.00	2,628.65	2,665.00	20.52	19.02	73.49%	**2,599.00**	2,619.25	2,659.00	14.56	24.91	73.46%
rec13+SD	**20,15**	4,337.00	3,343.00	3,374.10	3,422.00	21.40	26.77	77.08%	**3,319.00**	3,359.63	3,411.00	27.24	35.07	76.53%
rec15+SD	**20,15**	4,602.00	3,257.00	3,307.45	3,326.00	15.95	27.00	70.77%	**3,242.00**	3,279.03	3,316.00	22.96	35.90	70.45%
rec17+SD	**20,15**	4,390.00	**3,267.00**	3,296.75	3,320.00	18.78	27.34	74.42%	3,268.00	3,287.08	3,326.00	16.33	36.36	74.44%
rec19+SD	**30,10**	5,546.00	3,871.00	3,927.30	3,984.00	31.29	34.27	69.80%	**3,812.00**	3,886.10	3,987.00	34.57	45.57	68.73%
rec21+SD	**30,10**	5,033.00	3,781.00	3,804.90	3,851.00	19.96	33.05	75.12%	**3,729.00**	3,786.53	3,835.00	25.35	43.96	74.09%
rec23+SD	**30,10**	5,645.00	3,658.00	3,702.75	3,739.00	23.21	33.59	64.80%	**3,634.00**	3,682.30	3,730.00	24.93	44.67	64.38%
rec25+SD	**30,15**	6,456.00	4,666.00	4,676.40	4,689.00	6.28	45.83	72.27%	**4,659.00**	4,687.15	4,752.00	22.18	59.17	72.17%
rec27+SD	**30,15**	6,615.00	4,622.00	4,646.65	4,689.00	17.25	46.53	69.87%	**4,559.00**	4,608.18	4,650.00	25.65	60.06	68.92%
rec29+SD	**30,15**	6,339.00	4,458.00	4,493.25	4,538.00	20.73	46.38	70.33%	**4,452.00**	4,514.23	4,623.00	37.35	59.87	70.23%
rec31+SD	**50,10**	8,785.00	5,957.00	6,005.10	6,185.00	69.57	101.90	67.81%	**5,931.00**	6,029.98	6,096.00	42.44	131.55	67.51%
rec33+SD	**50,10**	8,970.00	6,183.00	6,231.20	6,310.00	33.65	101.70	68.93%	**6,178.00**	6,237.03	6,286.00	24.18	131.29	68.87%
rec35+SD	**50,10**	9,812.00	**6,148.00**	6,306.10	6,389.00	83.23	100.41	62.66%	6,169.00	6,242.55	6,351.00	44.46	129.63	62.87%
rec37+SD	**75,20**	15,066.00	**10,861.00**	10,954.20	11,011.00	37.50	633.81	72.09%	10,881.00	11,000.13	11,093.00	63.28	842.96	72.22%
rec39+SD	**75,20**	16,402.00	11,328.00	11,387.05	11,463.00	40.98	921.74	69.06%	**11,296.00**	11,415.30	11,510.00	53.37	1,015.75	68.87%
rec41+SD	**75,20**	16,952.00	11,450.00	11,536.85	11,598.00	35.91	634.87	67.54%	**11,441.00**	11,565.83	11,789.00	99.29	768.19	67.49%
Average	**NA**	**NA**	**NA**	**NA**	**NA**	**28.22**	**138.62**	**71.18%**	**NA**	**NA**	**NA**	**32.44**	**169.74**	**70.94%**

* Objective Function Value

**Standard Deviation

*** $\frac{Best\,OFV_{\text{Algorithm}}}{OFV_{\text{2-Opt}}} \times 100$, smaller gaps are more desirable

Table 12 – Comparison of the Results of Samarghandi and ElMekkawy (2014a) for $F \mid no-wait, S_{sd} \mid C_{\max}$ with the Results of the Developed Algorithms for $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$

Problem	n, m	No Server Constraint OFV*	GA - SR^2		GA - SR^3		GA+DLS - SR^2		GA+DLS - SR^3	
			OFV	Gap**	OFV	Gap	OFV	Gap	OFV	Gap
rec01+SD	**20,5**	2,139.00	2,124.00	99.29874	2,120.00	99.11173	2,118.00	99.01823	2,125.00	99.34549
rec03+SD	**20,5**	1,902.00	1,911.00	100.47319	1,890.00	99.36909	1,884.00	99.05363	1,880.00	98.84332
rec05+SD	**20,5**	2,028.00	2,007.00	98.96450	2,032.00	100.19724	2,010.00	99.11243	2,010.00	99.11243
rec07+SD	**20,10**	2,652.00	2,649.00	99.88688	2,671.00	100.71644	2,637.00	99.43439	2,671.00	100.71644
rec09+SD	**20,10**	2,657.00	2,660.00	100.11291	2,644.00	99.51073	2,675.00	100.67746	2,669.00	100.45164
rec11+SD	**20,10**	2,558.00	2,571.00	100.50821	2,600.00	101.64191	2,565.00	100.27365	2,599.00	101.60281
rec13+SD	**20,15**	3,309.00	3,315.00	100.18132	3,343.00	101.02750	3,324.00	100.45331	3,319.00	100.30221
rec15+SD	**20,15**	3,222.00	3,253.00	100.96214	3,257.00	101.08628	3,237.00	100.46555	3,242.00	100.62073
rec17+SD	**20,15**	3,271.00	3,274.00	100.09172	3,267.00	99.87771	3,271.00	100.00000	3,268.00	99.90828
rec19+SD	**30,10**	3,848.00	3,867.00	100.49376	3,871.00	100.59771	3,851.00	100.07796	3,812.00	99.06445
rec21+SD	**30,10**	3,756.00	3,743.00	99.65389	3,781.00	100.66560	3,714.00	98.88179	3,729.00	99.28115
rec23+SD	**30,10**	3,628.00	3,623.00	99.86218	3,658.00	100.82690	3,587.00	98.86990	3,634.00	100.16538
rec25+SD	**30,15**	4,654.00	4,662.00	100.17190	4,666.00	100.25784	4,644.00	99.78513	4,659.00	100.10743
rec27+SD	**30,15**	4,565.00	4,562.00	99.93428	4,622.00	101.24863	4,550.00	99.67141	4,559.00	99.86857
rec29+SD	**30,15**	4,422.00	4,443.00	100.47490	4,458.00	100.81411	4,424.00	100.04523	4,452.00	100.67843
rec31+SD	**50,10**	5,966.00	5,936.00	99.49715	5,957.00	99.84915	5,911.00	99.07811	5,931.00	99.41334
rec33+SD	**50,10**	6,186.00	6,159.00	99.56353	6,183.00	99.95150	6,155.00	99.49887	6,178.00	99.87068
rec35+SD	**50,10**	6,169.00	6,139.00	99.51370	6,148.00	99.65959	6,143.00	99.57854	6,169.00	100.00000
rec37+SD	**75,20**	10,782.00	10,985.00	101.88277	10,861.00	100.73270	10,957.00	101.62308	10,881.00	100.91820
rec39+SD	**75,20**	11,189.00	11,299.00	100.98311	11,328.00	101.24229	11,303.00	101.01886	11,296.00	100.95630
rec41+SD	**75,20**	11,324.00	11,494.00	101.50124	11,450.00	101.11268	11,461.00	101.20982	11,441.00	101.03320
Average	**NA**	**NA**	**NA**	**100.28248**	**NA**	**100.59481**	**NA**	**100.06922**	**NA**	**100.34411**

* Objective Function Value

** $\frac{OFV_{\text{Algorithm}}}{OFV_{\text{no server constraint}}} \times 100$, smaller gaps are more desirable

Results of table 12 are particularly remarkable for the following reasons:

- The difference between makespan of the problems with SR^2 constraints and problems with SR^3 constraints is infinitesimal.
- The difference between makespan of the problems with server constraints and makespan of the problems with no server constraints is very small.
- Solutions found for the problems with server constraints under SR^2 and SR^3 scenarios are in some cases better than the solutions of the same problem without server constraints.
- For the problems with optimal solutions, table 7 indicates that although the optimal solutions of some of the problems with server constraints are slightly larger than for the problems without server constraints, in many cases the optimal solutions are equal.

In other words, the server side constraints for the case of SR^2 and even SR^3 have either no effect or a negligible effect on the makespan of the problems studied in this research. This makes the results of this research very practical for companies that have adopted a lean approach as it may be possible to reduce the number of servers to $\frac{1}{2}$ (according to SR^2 scenario) or even $\frac{1}{3}$ (based on SR^3 scenario) with minimal impact on the makespan. Computational results of Samarghandi and ElMekkawy (2011) and Samarghandi and ElMekkawy (2013a) are in line with the computational results presented in section 5.

6. Conclusion

This paper considered the scheduling problem of $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$. The problem is strongly NP-Hard. A mathematical model of the problem was developed, and the problem was reduced to a permutation problem. The MCAS algorithm was developed to produce a feasible timetable for $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$ when a permutation of jobs is given. A genetic algorithm was developed to deal with the problem. A diversified local search sub-procedure was developed to further improve the computational results of the proposed GA and to increase the consistency of the solutions. A sensitivity analysis using ANOVA was performed to tune the parameters of the developed algorithms.

A thorough computational analysis was performed on the small- and large-instance test problems available in the literature. Computational analysis consisted of different server assignment scenarios. The developed algorithms proved to be very competitive; the algorithms were able to generate good-quality solutions for the test problems in a reasonable time. Computational results revealed that the impact of the server constraints on the makespan of the test problems were negligible. In fact, although the proposed methods were applied to test problems with server side constraints, they improved many of the best-known solutions proposed in the literature for problems without such constraints. These results are of importance for different settings where lean manufacturing techniques are practised.

A possibility for the future research is finding lower bounds for $F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$. Moreover, consideration of due dates for jobs and setting other objectives such as total or mean tardiness minimization is another future research direction. Also, considering sequence dependent setup times for the no-wait job shop problem is promising. Another important direction is to analytically define conditions for setup times that minimize the impact of the server constraints.

7. References

Aldowaisan, T. (2001). "A new heuristic and dominance relations for no-wait flowshops with setups." Computers & Operations Research **28**: 563-584.

Aldowaisan, T. and A. Allahverdi (1998). "Total flowtime in no-wait flowshops with separated setup times." Computers & Operations Research **25**: 757-765.

Aldowaisan, T. and A. Allahverdi (2004). "New heuristics for m-machine no-wait flowshop to minimize total completion time." Omega: 345-352.

Allahverdi, A., J. Gupta and T. Aldowaisan (1999). "A review of scheduling research involving setup considerations." Omega **27**: 219-239.

Araujoa, D. C. and M. S. Naganoa (2011). "A new effective heuristic method for the no-wait flowshop with sequence-dependent setup times problem." International Journal of Industrial Engineering Computations **2**: 155-166.

Beasley, J. E. (July 2009). "OR-Library: distributing test problems by electronic mail." Retrieved March, 2014, from http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

Bianco, L., P. Dell'Olmo and S. Giordani (1999). "Flow shop no-wait scheduling with sequence dependent setup times and release dates." INFOR **37**(1): 3-20.

Bonney, M. and S. Gundry (1976). "Solutions to the constrained flowshop sequencing problem." Operational Research Quarterly **24**: 869-883.

Carlier, J. (1978). "Ordonnancements a contraintes disjonctives." RAIRO Recherche Operationnelle **12**: 333-351.

Cheng, T. C. E., G. Wang and C. Sriskandarajah (1999). "One-operator two-machine flowshop scheduling with setup and dismounting times." Computers and Operations Research **26**: 715-730.

Gangadharan, R. and C. Rajendran (1993). "Heuristic algorithms for scheduling in no-wait flowshop." International Journal of Production Economics **32**: 285-290.

Gao, K.-Z., Q.-K. Pan, J.-Q. Li, Y.-T. Wang and J. Liang (2012). "A hybrid harmony search algorithm for the no-wait flow-shop scheduling problems." Asia-Pacific Journal of Operational Research **29**(2).

Glass, C. A., J. Gupta and C. Potts (1999). "Two-machine no-wait flow shop scheduling with missing operations." Mathematics of Operations Research **24**(4): 911-924.

Grabowski, J. and J. Pempera (2000). "Sequencing of jobs in some production systems." European Journal of Operational Research **125**: 535-550.

Grabowski, J. and J. Pempera (2005). "Some local search algorithms for no-wait flow-shop problem with makespan criterion." Computers & Operations Research **32**: 2197–2212.

Grabowski, J. and M. Syslo (1973). "On some machine Sequencing problems (I)." Applications of Mathematicae **13**: 340-345.

Guirchoun, S., P. Martineau and J. C. Billaut (2005). "Total completion time minimization in a computer system with a server and two parallel processors." Computers & Operations Research **32**: 599-611.

Gupta, J. N. D., V. A. Strusevich and C. M. Zwaneveld (1997). "Two-stage no-wait scheduling models with setup and removal times separated." Computers & Operations Research **24**(1): 1025-1031.

Hall, N. and C. Sriskandarajah (1996). "A survey of machine scheduling problems with blocking and no-wait in process." Operations Research **44**: 510-525.

Jolai, F., M. Rabiee and H. Asefi (Article in Press). "A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times." International Journal of Production Research.

King, J. and A. Spachis (1980). "Heuristics for flowshop scheduling." International Journal of Production Research **18**: 343-357.

Laha, D. and U. K. Chakraborty (2009). "A constructive heuristic for minimizing makespan in no-wait flow shop scheduling." International Journal of Advanced Manufacturing Technology **41**: 97-109.

Liu, B., L. Wang and Y.-H. Jin (2007). "An effective hybrid particle swarm optimization for no-wait flow shop scheduling." International Journal of Advanced Manufacturing Technology **31**: 1001-1011.

Nagano, M. S., A. A. d. Silva and L. A. N. Lorena (Article in Press). "A new evolutionary clustering search for a no-wait flowshop problem with set-up times." Engineering Applications of Artificial Intelligence.

Pan, Q.-K., L. Wang, M. F. Tasgetiren and B.-H. Zhao (2008a). "A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion." International Journal of Advanced Manufacturing Technology **38**: 337 - 347.

Pan, Q.-K., L. Wang and B.-H. Zhao (2008b). "An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion." International Journal of Advanced Manufacturing Technology **38**(7-8): 778-786.

Qian, B., L. Wang, R. Hu, D. X. Huang and X. Wang (2009). "A DE-based approach to no-wait flow-shop scheduling." Computers & Industrial Engineering **57**: 787-805.

Raaymakers, W. and J. Hoogeveen (2000). "Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing." European Journal of Operational Research **126**: 131-151.

Rabiee, M., M. Zandieh and A. Jafarian (Article in Press). "Scheduling of a no-wait two-machine flow shop with sequence-dependent setup times and probable rework using robust meta-heuristics." International Journal of Production Research.

Rajendran, C. (1994). "A no-wait flowshop scheduling heuristic to minimize makespan." Journal of the Operational Research Society **45**: 472-478.

Reddi, S. and C. Ramamoorthy (1972). "On the flowshop sequencing problem with no-wait in process." Operational Research Quarterly **23**: 323-331.

Reeves, C. (1995). "A genetic algorithm for flowshop sequencing." Computers and Operations Research **22**: 5-13.

Röck, H. (1984). "Some new results in flow shop scheduling." Zeitschrift für Operations Research **28**: 1-16.

Samarghandi, H. and T. Y. ElMekkawy (2011). "An effective hybrid algorithm for the two-machine no-wait flow shop problem with separable setup times and single server." European Journal of Industrial Engineering **5**(2): 111-131.

Samarghandi, H. and T. Y. ElMekkawy (2012a). "A meta-heuristic approach for solving the no-wait flow shop problem." International Journal of Production Research **50**(24): 7313 - 7326.

Samarghandi, H. and T. Y. ElMekkawy (2012b). "A Metaheuristic approach for the no-wait flow shop problem with separable setup times and makespan criterion." International Journal of Production Research **61**(9-12): 1101 - 1114.

Samarghandi, H. and T. Y. ElMekkawy (2013a). "Two-machine no-wait job shop problem with separable setup times and single-server constraints." International Journal of Advanced Manufacturing Technology **61**(1-4): 295 - 308.

Samarghandi, H. and T. Y. ElMekkawy (2013b). "Two-machine, no-wait job shop problem with separable setup times and single-server constraints." The International Journal of Advanced Manufacturing Technology **65**(1-4): 295-308.

Samarghandi, H. and T. Y. ElMekkawy (2014a). "Solving the no-wait flow-shop problem with sequence-dependent set-up times." International Journal of Computer Integrated Manufacturing **27**(3): 213 - 228.

Samarghandi, H. and T. Y. ElMekkawy (2014b). "Solving the no-wait flow-shop problem with sequence-dependent set-up times." International Journal of Computer Integrated Manufacturing **27**(3): 213-228.

Samarghandi, H. and K. Eshghi (2009). An efficient tabu algorithm for solving the single row facility layout problem. Computers & Industrial Engineering, 2009. CIE 2009. International Conference on, IEEE.

Samarghandi, H. and F. F. Jahantigh (2011). "A particle swarm optimisation for fuzzy dynamic facility layout problem." International Journal of Metaheuristics **1**(3): 257-278.

Samarghandi, H., P. Taabayan and F. F. Jahantigh (2010). "A particle swarm optimization for the single row facility layout problem." Computers & Industrial Engineering **58**(4): 529-534.

Shadrokh, S. and F. Kianfar (2007). "A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty." European Journal of Operational Research **181**: 86-101.

Sidney, J., C. Potts and C. Sriskandarayah (2000). "Heuristic for scheduling two-machine no-wait flow shops with anticipatory setups." Operations Research Letters **26**(4): 165-173.

Su, L. H. and Y. Y. Lee (2008). "The two-machine flowshop no-wait scheduling problem with a single server to minimize the total completion time." Computers & Operations Research **35**: 2952-2963.

Sviridenko, M. (2003). "Makespan minimization in no-wait flow shops: A polynomial time approximation scheme." SIAM Journal on Discrete Mathematics **16**(2): 313-322.

Wismer, D. (1972). "Solution of the flowshop scheduling with no intermediate queues." Operations Research **20**: 689-697.

Ying, K.-C., Z.-J. Lee, C.-C. Lu and S.-W. Lin (2012). "Metaheuristics for scheduling a no-wait flowshop manufacturing cell with sequence-dependent family setups " The International Journal of Advanced Manufacturing Technology **58**(5-8): 671-682.